Visual Recognition Using Directional Distribution Distance

Jianxin Wu Bin-Bin Gao
National Key Laboratory for Novel Software Technology
Nanjing University, China
wujx2001@nju.edu.cn, gaobb@lamda.nju.edu.cn

Guoqing Liu
Minieye, Youjia Innovation LLC
guoqing@minieye.cc

Abstract

In computer vision, an entity such as an image or video is often represented as a set of instance vectors, which can be SIFT, motion, or deep learning feature vectors extracted from different parts of that entity. Thus, it is essential to design efficient and effective methods to compare two sets of instance vectors. Existing methods such as FV, VLAD or Super Vectors have achieved excellent results. However, this paper shows that these methods are designed based on a generative perspective, and a discriminative method can be more effective in categorizing images or videos. The proposed D3 (discriminative distribution distance) method effectively compares two sets as two distributions, and proposes a directional total variation distance (DTVD) to measure how separated are they. Furthermore, a robust classifier-based method is proposed to estimate DTVD robustly. The D3 method is evaluated in action and image recognition tasks and has achieved excellent accuracy and speed. D3 also has a synergy with FV. The combination of D3 and FV has advantages over D3, FV, and VLAD.

1. Introduction

In visual recognition, an entity (object or video) is usually represented as a set of instance vectors. Each instance vector is extracted using part of the entity (*e.g.*, a local window extracted from an image or a time-space subvolume extracted from a video). Various features have emerged as the state-of-the-art to extract instance vectors at different stages of recognition research, such as dense SIFT features [21], dense CENTRIST features [29] or CNN features for images [13], or (improved) dense trajectory features [28] or CNN features for videos [7]. Although originally CNN (or other deep learning methods) integrates visual representation and classification into one system [19, 15], recent works have shown that if multiple (a set of) CNN features are extracted from entities and classify images or videos based on these sets, higher accuracies can be obtained [8, 32, 3, 31].

Because most existing learning algorithms assume that an entity is represented as a vector instead of a set of vectors, we need to find a suitable visual representation that encodes the set of instance vectors into one single vector. It is desirable that the representation will capture useful (*i.e.*, discriminative) information from the set. Thus, comparing one entity (a set of instance vectors) to another can be divided into two steps: first represent the sets as two vectors, then find a suitable distance metric to compare the vectors. One useful variant is to compare one entity to a set of entities (*e.g.*, all training images, corresponding to a bigger union set by gathering the instance vectors in every image), which is often used too.

Since the ℓ_2 distance (or correspondingly linear SVM) is very efficient and has shown great accuracy in the second step, an effective visual representation that turns a set of instance vectors into one single vector (*i.e.*, the first step) has been very important in visual recognition research efforts. Many representations have been proposed, for example,

- **Fisher Vector (FV) and VLAD.** FV [25] is based on the idea of Fisher kernel in machine learning [11]. It models the distribution of instance vectors in training entities using a Gaussian Mixture Model (GMM). Then, one training or testing entity is modeled generatively, by a vector which describes how the GMM can be modified to generate the instance vectors inside that entity. A GMM with K components has three sets of parameters $(w_i, \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$, $1 \le i \le K$. VLAD [12], another popular visual representation, can be regarded as a special case of FV, by using only the $\boldsymbol{\mu}$ parameters. The classic bag-of-visual-words (BOVW) [4] representation is also a special case of FV, using the w parameters.
- **Super-Vector** Instead of modeling the instance vectors as distributions, the Super-Vector [35] represents a set of instance vectors based on how they can be reconstructed from dictionary items. Super-Vector aims at reducing the reconstruction error, which is also from a generative perspective. The output of Super-Vector has two parts, which are conceptually related to the w and $\boldsymbol{\mu}$ parame-

ters in Fisher Vectors.

Both threads of methods have shown excellent accuracy in the literature. However, they both focus on modeling how one entity or one distribution is *generated*. Given the fact that the task in hand is recognition, we argue that *we need to pay more attention to how two entities or two distributions are separated.* In other words, we need a visual representation that pays more attention to the discriminative side. We naturally expect that such a representation would be suitable for visual recognition tasks, whose objective is to properly separate entities belonging to different categories.

In this paper, we propose a discriminative distribution distance (D3) representation that converts a set of instance vectors into a vector representation. D3 explicitly considers two distributions: a density X which is estimated from the training set as a reference model, and one entity forms another distribution Y. D3 then uses the distribution distance between X and Y as a discriminative representation for the entity Y. Technically, D3 has the following contributions.

- We propose a direction total variation distance (DTVD) to measure the distance between X and Y, which contains more discriminative information than classic distribution distances by considering *directions*;
- Directly calculating DTVD is unstable and problematic because Y may be non-Gaussian and only contains few items. We propose to estimate DTVD in a discriminative manner, by *calculating robust classification errors* when we try to classify every dimension of X from Y;
- We also show that D3 and FV are complementary to each other. By combining D3 and FV, we can achieve an accuracy higher than D3, FV, and VLAD.

We will start by explaining closely related methods, then proposing the directional distribution distance, its robust estimation, and the entire D3 pipeline in Sec. 2. Sec. 3 presents empirical results, and Sec. 4 concludes this paper.

2. Discriminative Distribution Distance

In this section, we propose a discriminative distribution distance (abbreviated as D3) to compare two sets of observations, which leads to an efficient and effective visual representation.

2.1. Distribution distance: generative vs. discriminative

Given two objects X and Y, each of which is represented as an unordered set of instance vectors, *i.e.*, $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_X}\}$, $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_Y}\}$, we are interested in finding $d(X, Y)$, the distance (or dissimilarity) between them. This task is frequently encountered in compute vision. For example, an image or a video is usually represented as a set of feature vectors extracted from various image patches or supervoxels.

In the Fisher Vector (FV) representation, a large set of instance vectors are extracted from training images or videos. We treat this set as X and a Gaussian Mixture Model (GMM) p_X with parameters $\lambda = \left\{(w_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right\}_{k=1}^{K}$ is estimated from X. When a test image or video $\mathcal{Y}$ is presented, we extract its instance vectors and treat it as Y. The FV representation considers X and Y as generated from two underlying distributions p_X and p_Y, and encodes $\mathcal{Y}$ as a vector $\boldsymbol{f}$. This is a generative model and each component in $\boldsymbol{f}$ describes how each parameter in λ should be modified such that p_X can be modified to fit the data Y properly.

Specifically, the probability that $\boldsymbol{y}_i$ is generated by the k-th Gaussian is

$$\gamma_i(k) = p(k|\boldsymbol{y}_i, \lambda) = \frac{1}{Z} w_k p_k(\boldsymbol{y}_i|\lambda)\,, \tag{1}$$

where Z is a normalization constant, p_k is the k-th Gaussian component with weight w_k, and parameters $(\boldsymbol{\mu}_k, \Sigma_k)$. In FV, the GMM covariances Σ_k are assumed to be diagonal, whose diagonal entries form a vector $\boldsymbol{\sigma}_k$. The trends of parameter changing (gradients) that modify p_X to fit $\mathcal{Y}$ is then (for all $1 \le k \le K$) [25]

$$\boldsymbol{f}_{w_k} = \frac{1}{\sqrt{w_k}} \sum_{i=1}^{n_Y} \left(\gamma_i(k) - w_k\right)\,, \tag{2}$$

$$\boldsymbol{f}_{\boldsymbol{\mu}_k} = \frac{1}{\sqrt{w_k}} \sum_{i=1}^{n_Y} \gamma_i(k) \left(\frac{\boldsymbol{y}_i - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k}\right)\,, \tag{3}$$

$$\boldsymbol{f}_{\boldsymbol{\sigma}_k} = \frac{1}{\sqrt{2w_k}} \sum_{i=1}^{n_Y} \gamma_i(k) \left(\frac{(\boldsymbol{y}_i - \boldsymbol{\mu}_k)^2}{(\boldsymbol{\sigma}_k)^2} - 1\right)\,, \tag{4}$$

which correspond to the $w, \boldsymbol{\mu}, \boldsymbol{\sigma}$ parameters for p_X, respectively. The image or video $\mathcal{Y}$ is then represented by a feature vector $\boldsymbol{f}$, which concatenates $\boldsymbol{f}_{w_k}$, $\boldsymbol{f}_{\boldsymbol{\mu}_k}$, and $\boldsymbol{f}_{\boldsymbol{\sigma}_k}$ for all $1 \le k \le K$.

We want to emphasize two observations based on the FV representation.

- The gradient vector $\boldsymbol{f}$ is formed under the *generative* assumption that Y can be modeled by p_X if we are allowed to modify the parameter set λ. Since what we are really interested in is how far is X from Y, we believe that a *discriminative* distance between X and Y is a better option. That is, in this paper we will treat X and Y as sampled from two different distributions p_X and p_Y, and find their distribution distance to encode the image or video $\mathcal{Y}$;
- Since diagonal Σ_k are used, after the soft assignment probabilities $\gamma_i(k)$ are calculated, each dimension of $\boldsymbol{f}$ is generated independent of any other dimension. Thus, in finding a suitable representation for Y, we only need to consider each dimension individually. The problem is then: given two sets of *scalar* values $X =$

$\{x_1, x_2, \ldots, \}$ and $Y = \{y_1, y_2, \ldots\}$ (sampled from 1-d distribution p_X and p_Y, respectively), how do we properly compute $d(p_X, p_Y)$?

As a final note in this section, the VLAD and super vector representation can be interpreted as special cases of FV, while VLAD uses the $\boldsymbol{f}_{\boldsymbol{\mu}_k}$ components of $\boldsymbol{f}$, and super vectors use both $\boldsymbol{f}_{w_k}$ and $\boldsymbol{f}_{\boldsymbol{\mu}_k}$. It is also a common practice to use only $\boldsymbol{f}_{\boldsymbol{\mu}_k}$ and $\boldsymbol{f}_{\boldsymbol{\sigma}_k}$ in FV implementations.

2.2. Directional Total Variation Distance

We need to be more discriminative. Thus, we propose to explicitly consider two distributions X and Y, where X is a density of instance vectors estimated from the training set, and Y is from one (training or testing) entity. A representation of Y that encodes the *distance* between X and Y will contain useful discriminative information about Y.

A widely used distance that compares two distributions is the *total variation* distance, which is independent of the distributions' parameterizations. Let ν_1 and ν_2 be two probability measures on a measurable space $(\mathcal{O}, \mathcal{B})$, the total variation distance is defined as

$$d_{TV}(\nu_X, \nu_Y) \triangleq \sup_{A \in \mathcal{B}} |\nu_X(A) - \nu_Y(A)| \,. \tag{5}$$

While this definition is rather formal, d_{TV} has a more intuitive equation for commonly used continuous distributions by the Scheffe's Lemma [5]. For example, for two normal distributions with p.d.f. $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$,

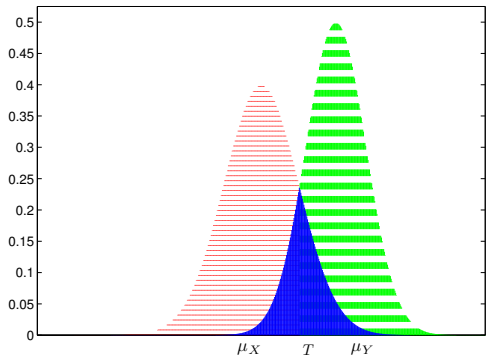
$$d_{TV}(p_X, p_Y) = \frac{1}{2} \int_u |p_X(u) - p_Y(u)| \, \mathrm{d}u \,. \tag{6}$$

As illustrated in Fig. 1a, it is half the summed area of the red and green regions, which clearly indicates how two distributions are separated from each other.

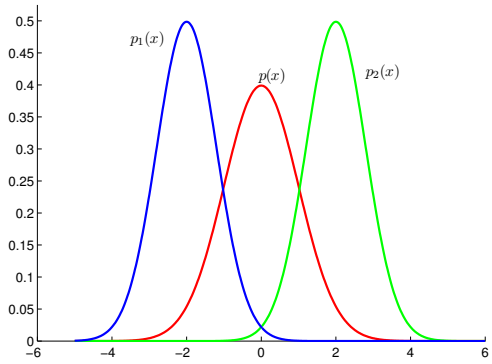
The classic total variation distance (Eq. 6), however, is missing one most important information that captures the key difference between p_X and p_Y, as shown in Fig. 1b. In Fig. 1b, p_1 and p_2 are symmetric with respect to the mean of p, thus we have $d_{TV}(p, p_1) = d_{TV}(p, p_2)$, in spite of the fact that p_1 and p_2 are far apart. The missing of directional information is responsible for this drawback. Thus, we propose a directional total variation distance (DTVD) as

$$d_{DTV}(p_X, p_Y) = \mathrm{sign}(\mu_Y - \mu_X) \times d_{TV}(p_X, p_Y) \,. \tag{7}$$

DTVD is a signed distance. In Fig. 1b, we will (correctly) have $d_{DTV}(p, p_1) = -d_{DTV}(p, p_2)$, which clearly signifies the difference between p_1 and p_2. Obviously the distance function d_{DTV} is not a metric, because it is neither non-negative, nor symmetric.



(a)



(b)

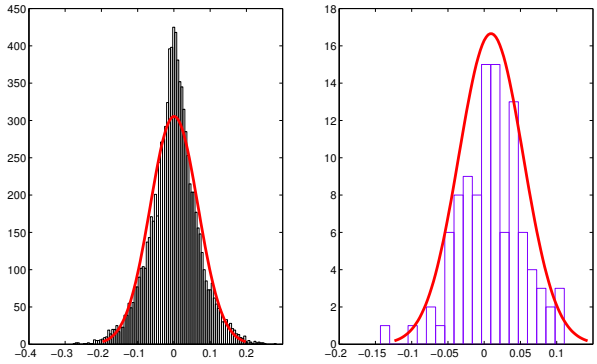
Figure 1. Illustration of the total variation distance. 1a illustrates d_{TV} for two Gaussians, and 1b reveals that *direction* is essential.

2.3. Robust estimation of the DTVD

For two Gaussians p_X and p_Y, their p.d.f. will have two intersections if $\sigma_X \neq \sigma_Y$. For example, in Fig. 1a the second intersection is in the far right end of the x-axis. A closed-form solution to calculate d_{TV} based on both intersections is available [5]. However, this closed-form solution leads to serious performance drop when used in visual recognition in our experiments. We conjecture that two reasons have caused this issue:

- The distributions are not necessarily normal. As shown in Fig. 2, the typical example of p_X (in Fig. 2a) is generated from many training instances, its shape resembles that of a Gaussian, but has a shaper peak. p_Y, which is generated from a single image, deviates from a normal distribution;
- Since the set Y (which is extracted from a single image or video, cf. Fig. 2b) usually contains small number of instance vectors, this fact leads to unstable estimation of its distribution parameters, and hence unstable $d_{DTV}(p_X, p_Y)$. Thus, we need a more robust way to estimate the distribution distance.

Our key insight again arises from the discriminative perspective. It is obvious that the total variation distance d_{TV}



(a) (b)

Figure 2. Typical distribution of feature values. 2a is calculated based on features used to generate the codebook, and 2b is from a single image. The red curve is a normal distribution estimated from the same data. This figure is generated using bag of dense SIFT on the Scene 15 dataset with $K = 64$ VLAD encoding. The dimension shown is the 37-th dimension in the 37-th cluster of the codebook.

is equivalent to one minus the *Bayes error* of a binary classification problem, where the two classes have equal prior and follow p_X and p_Y, respectively. Thus, we can estimate d_{TV} (hence d_{DTV}) by *robustly estimating the classification error between the two sets of examples X and Y*. Note that this task is easy since X and Y only contain scalar examples.

We adopt the minimax probability machine (MPM) [17] to estimate the classification error. MPM is robust because it minimizes the maximum probability of misclassification, hence the name minimax. Given examples X with mean $\boldsymbol{\mu}_X$ and covariance Σ_X and examples Y with $\boldsymbol{\mu}_Y$ and Σ_Y, the classifier boundary $\boldsymbol{a}^T\boldsymbol{x} - b = 0$ is determined by the MPM problem

$$\kappa_\star^{-1} \triangleq \min_{\boldsymbol{a}} \sqrt{\boldsymbol{a}^T\Sigma_X\boldsymbol{a}} + \sqrt{\boldsymbol{a}^T\Sigma_Y\boldsymbol{a}} \tag{8}$$

$$\text{s.t.} \quad \boldsymbol{a}^T(\boldsymbol{\mu}_X - \boldsymbol{\mu}_Y) = 1\,, \tag{9}$$

and

$$b_\star = \boldsymbol{a}_\star^T\boldsymbol{\mu}_X - \kappa_\star\sqrt{\boldsymbol{a}_\star^T\Sigma_X\boldsymbol{a}_\star}\,. \tag{10}$$

Eq. 9 is a second order cone problem (SOCP) that can be solved by an iterative algorithm. However, since we are dealing with scalar examples that (assumed to) follow normal distributions, it has a closed form solution. Note that $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$, we can immediately get the following boundary $a_\star x - b_\star = 0$, where

$$a_\star = \frac{1}{\mu_X - \mu_Y}\,, \quad b_\star = a_\star \times \frac{\mu_X\sigma_Y + \mu_Y\sigma_X}{\sigma_X + \sigma_Y}\,, \tag{11}$$

$$\kappa_\star = \frac{|\mu_X - \mu_Y|}{\sigma_X + \sigma_Y}\,. \tag{12}$$

That is, the two 1-d distributions p_X and p_Y are classified at the threshold value

$$T = \frac{\mu_X\sigma_Y + \mu_Y\sigma_X}{\sigma_X + \sigma_Y}\,. \tag{13}$$

If we re-use Fig. 1a and (approximately) assume the red, blue, and green areas intersect at $T = \frac{\mu_X\sigma_Y + \mu_Y\sigma_X}{\sigma_X + \sigma_Y}$, which is guaranteed to reside in between μ_X and μ_Y. Then, the area of the blue region is:

$$Area = 1 - \Phi\left(\frac{T - \mu_X}{\sigma_X}\right) + \Phi\left(\frac{T - \mu_Y}{\sigma_Y}\right)\,. \tag{14}$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}\,\mathrm{d}t \tag{15}$$

is the cumulative distribution function (c.d.f.) of a standard normal distribution $N(0, 1)$. And, we have

$$d_{DTV}(p_X, p_Y) = 2 - 2Area = 4\Phi\left(\frac{\mu_Y - \mu_X}{\sigma_X + \sigma_Y}\right) - 2\,, \tag{16}$$

making use of the fact that

$$\frac{T - \mu_X}{\sigma_X} = \frac{\mu_Y - \mu_X}{\sigma_X + \sigma_Y} = -\frac{T - \mu_Y}{\sigma_Y},$$

and the property of Φ that $\Phi(-x) = 1 - \Phi(x)$.

Two points are worth mentioning about Eq. 16.

- Although our derivation and Fig. 1a is assuming $\mu_X < \mu_Y$, it is easy to derive that when $\mu_X \geq \mu_Y$, Eq. 16 still holds. And, it always have the same sign as $\mu_Y - \mu_X$. Hence, Eq. 16 computes d_{DTV} instead of d_{TV}, and its range is $[-2\ \ 2]$.
- In practice we use the error function. The error function is defined as

$$\mathrm{erf}(x) = \frac{1}{\sqrt{\pi}}\int_{-x}^{x} e^{-t^2}\,\mathrm{d}t\,, \tag{17}$$

and it satisfies that

$$\Phi(x) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)\,. \tag{18}$$

Thus, we have

$$d_{DTV}(p_X, p_Y) = 2\,\mathrm{erf}\left(\frac{\mu_Y - \mu_X}{\sqrt{2}(\sigma_X + \sigma_Y)}\right)\,. \tag{19}$$

The error function erf is built-in and efficient in most major programming languages, which facilitates the calculation of d_{DTV} using Eq. 19.

We also want to note there has been research to model the discriminative distance between two sets of instance vectors. In [23], non-parametric kernels are estimated from two

Algorithm 1 Visual representation using D3

Input: An image or video $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots\}$; and, a dictionary (visual code book) with size K and cluster mean $\boldsymbol{\mu}_k$ and standard deviations $\boldsymbol{\sigma}_k$ $(1 \leq k \leq K)$
for $i = 1, 2, \ldots, K$ **do**
  $Y' = \{\boldsymbol{y}_j | \boldsymbol{y}_j \in Y, \arg\min_{1 \leq k \leq K} \|\boldsymbol{y}_j - \boldsymbol{\mu}_k\| = i\}$
  Compute the mean and standard deviation vectors of the set Y', denote as $\boldsymbol{\mu}'$ and $\boldsymbol{\sigma}'$, respectively
  $\boldsymbol{f}_i = \text{erf}\left(\frac{\boldsymbol{\mu}' - \boldsymbol{\mu}_i}{\sqrt{2}(\boldsymbol{\sigma}' + \boldsymbol{\sigma}_i)}\right)$.
  Note that the erf function is applied to every component of the $\frac{\boldsymbol{\mu}' - \boldsymbol{\mu}_i}{\sqrt{2}(\boldsymbol{\sigma}' + \boldsymbol{\sigma}_i)}$ vector individually
  $\boldsymbol{f}_i = \frac{\boldsymbol{f}_i}{\|\boldsymbol{f}_i\|}$
end for
$\boldsymbol{f} = [\boldsymbol{f}_1^T \ \boldsymbol{f}_2^T \ \ldots \ \boldsymbol{f}_K^T]$
$\boldsymbol{f} = \frac{\boldsymbol{f}}{\|\boldsymbol{f}\|}$
Output: The new representation $\boldsymbol{f} \in \mathbb{R}^{d \times K}$

sets, and use the Hellinger's distance or the Rényi-α divergence to measure the distance between two distributions. This method, however, suffers from one major limitation. Non-parametric kernel estimation is very time consuming, which took 3.3 days in a subset of the Scene 15 dataset, a fact that renders it impractical for large problems. As a direct comparison, D3 only requires less than 2 minutes.

2.4. The pipeline using d_{DTV} for visual recognition

We assume that an image or video $\mathcal{Y}$ is represented as a bag of instance feature vectors $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots\}$, where each $\boldsymbol{y}_i \in \mathbb{R}^d$. The instance vectors are usually extracted as dense SIFT vectors or deep learning (CNN) features for images, or dense trajectory features or deep learning features for videos, or other representations that use a set of vectors to represent an entity.

The pipeline to use d_{DTV} to generate image or video representation follows two steps.

- **Dictionary generation.** For simplicity and computational efficiency, we collect a large set of instance vectors from the training set, and then use the k-means algorithm to generate a dictionary that partitions the space of instance vectors into K regions. We compute the mean and standard deviation of the instance vectors inside cluster k as $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$ for all $1 \leq k \leq K$. Values in the standard deviation vector $\boldsymbol{\sigma}_k$ is computed for every dimension independently;
- **Visual representation.** Given an image or video $\mathcal{Y}$, we use Algorithm 1 to convert it to a vector representation. Note that since we normalize every $\boldsymbol{f}_i$ in Algorithm 1, the constant factor ('2') in Eq. 19 is not necessary and is thus omitted.

In Algorithm 1, we use the k-means algorithm to generate a visual codebook, and an instance vector is hard-assigned to one visual code word. A GMM model can also be used as a soft codebook, similar to what is performed in FV. However, a GMM has higher costs in generating both the dictionary and visual representation. Thus, we use k-means to generate a codebook in D3. Then, D3 and VLAD have very similar frameworks, and it is interesting to compare D3 with both VLAD and FV.

2.5. Efficiency and hybrid representation

Since the error function implementation is efficient, the computational cost of D3 is roughly the same as that of VLAD, which is much more efficient than the FV method. The evaluation in [22] showed that the time for VLAD is only less than 5% of that of FV. Thus, a visual representation using D3 is efficient to compute.

It is also worth noting that although D3 and FV both used first- and second-order statistics of an image or video Y and compare these statistics with those computed from the training set, they use these statistics in very different ways. Thus, different information (discriminative vs. generative) are extracted by D3 and FV. By computing the D3 and FV representation separately and then concatenate them together to form a hybrid one, we can get higher recognition accuracy than both D3 and FV, as will be shown in Sec. 3. Suppose we form a dK dimensional D3 vector and a dK dimensional FV vector, the hybrid representation will be $2dK$ dimensional. However, its computational time will be only roughly half of that of forming a $2dK$ dimensional FV representation.

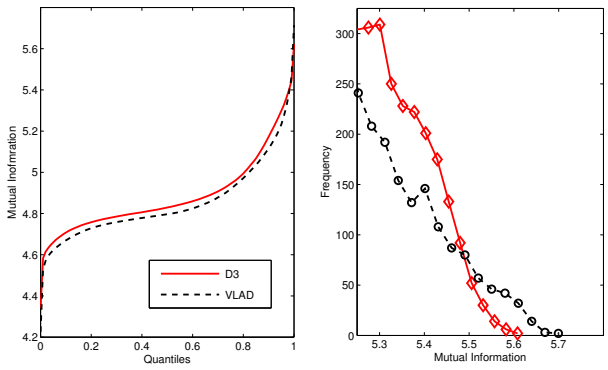
A final note is about higher order VLAD. VLAD only uses first-order statistics (mean) of the set of instance vectors. In [22], higher-order statistics (variance and skewness) are added to effectively improve VLAD. Since this method will triple the number of dimensions of VLAD (with the same K) and its accuracy is not as high as FV, we will not empirically compare D3 with this method in this paper. However, because D3 does not specify how a codebook is generated, the supervised codebook generation method of [22] can be adopted to further improve D3 in the future.

3. Experimental Results

To compare the representations fairly, we compare them using the same number of dimensions. For example, the following setups will be compared to each other.

- D3 (or VLAD) with $K_1 = 256$ visual words; the representation has dK_1 dimensions;
- FV with $K_2 = 128$ components ($2dK_2 = dK_1$);
- A mixture of D3 and FV with $K_3 = 128$ in D3 and $K_4 = 64$ in FV ($dK_3 + 2dK_4 = 2dK_2 = dK_1$).

We will use D3's K size to indicate the size of all the above setups (*i.e.*, $K = 256$ in this example).



(a) (b)

Figure 3. Distribution of per-dimensional mutual information. 3a shows the quantile values in the full range, and 3b is the frequency of high (most discriminative) MI values.

Three types of experiments are performed. First, a small image dataset is used to study the property of D3 (Sec. 3.1). Then, D3 is evaluated in action recognition (with the ITF features, in Sec. 3.2) and in image recognition (with CNN features, in Sec. 3.3). Discussions are in Sec. 3.4.

3.1. Why use D3?

We first study the properties of the proposed D3 representation, and shed some lights on why it is an effective way to encode the distance between two sets of dense SIFT instance vectors.

Using the training images of the Scene 15 dataset [18] and dense SIFT features (with step size 4), we compare the per-dimensional discriminative power of these two representations (D3 and VLAD).

Suppose X is the D3 or VLAD representation of a set of images with corresponding image labels $\boldsymbol{l}$, whose i-th dimension form a vector $\boldsymbol{x}_{:i}$. It is natural to measure the discriminative power of the i-th dimension by computing the mutual information between $\boldsymbol{x}_{:i}$ and $\boldsymbol{l}$, *i.e.*, $\mathrm{MI}(\boldsymbol{x}_{:i}, \boldsymbol{l})$ [33]. We use the 2-bit method in [33] to quantize $\boldsymbol{x}_{:i}$ and compute the mutual information. The distribution of all dimension's MI values are shown in Fig. 3.

Fig. 3a shows the MI's quantile values. For example, when the x-axis is 0.5, the D3 and VLAD curve has value 4.8292 and 4.7966, meaning that the median of D3's MI value is above the of VLAD's by 0.0326. Similarly, when the y-axis is 4.8282, the D3 and VLAD curves has corresponding quantile (x-axis) values 0.5 and 0.3888, meaning that 50% of D3's MI value is above 0.4282, but only 38.88% of VLAD's dimensions reaches this discriminative power. Since the D3 (red solid) curve is almost consistently above the VLAD (dashed black) curve, D3's dimensions have higher discriminative power than VLAD's.

Fig. 3b shows the frequencies of dimensions that have the highest MI values. Although VLAD has a few dimensions that have higher MI values than D3, D3 obviously have many more discriminative dimensions. The total number of dimensions with MI values > 5.25 are 2020 and 1544 for D3 and VLAD, respectively, a 30.8% advantage for D3. Since every single dimension is too weak to classify the image, it is more important to have many dimensions with good discriminative powers than having few only slightly more discriminative ones.

3.2. Action recognition results

We first show experimental results for action recognition. A set of improved trajectory features (ITF) [28] are extracted and then converted to D3, VLAD, FV, and two hybrid representations (D3+FV and VLAD+FV). The default parameters are used to extract ITF features.

We experimented on three datasets: UCF 101 [27], HMDB 51 [16] and Youtube [20]. For UCF 101, the three splits of train and test videos in [14] are used and we report the average accuracy. This dataset has 13320 videos and 101 action categories. The HMDB 51 dataset has 51 actions in 6766 clips. We use the original (not stabilized) videos and follow [16] to report average accuracy of its 3 predefined splits of training / testing videos. Youtube is a small scale dataset with 11 action types. There are 25 groups in each action category and 4 videos are used in each group. Following the original protocol, we report the average of the 25-fold leave one group cross validation accuracy rates. Results on these datasets are reported in Table 1. We summarize the experimental results into the following observations.

D3 is better than VLAD in almost all cases. In the 12 comparisons between D3 and VLAD, D3 wins in 11 cases. D3 often has a margin even if it use half of number of dimensions of VLAD (*e.g.*, D3 $K = 128$ vs. VLAD $K = 256$). It shows that the D3 representation is effective in capturing useful information for classification.

D3 bridges the gap between VLAD and FV. In practice we often see that FV has higher accuracy than VLAD, but also much higher computational costs. D3 has roughly the same speed as VLAD, but its accuracy is close to that of FV. Compared to VLAD whose accuracies are usually 2–3% lower than FV, D3 has much closer accuracy rates to FV. On average, D3 is 1% worse than FV. On the Youtube dataset D3 is better than FV (91.55% vs. 91.00%). Given the computational benefits of D3, it can act as an attractive alternative for FV.

The hybrid D3 / FV representation (nearly) consistently outperforms all other methods. We show that the hybrid methods are the best performers in Table 1. The D3+FV representation is especially effective: it is the winner in 8 out of 9 cases. With $K = 128$ in the Youtube set being the only exception, D3+FV consistently beats other methods,

Table 1. Action recognition accuracy (%) comparisons. Note that the results in one column are compared with the same number of dimensions in the representations. For example, the column K=256 means that $K = 256$ for D3 and VLAD, $K = 128$ for FV, and in the hybrid representation, $K = 128$ for D3 or VLAD combined with $K = 64$ for FV. Note that $K = 64$ results for the hybrid representation is not presented. The best results are shown in bold face.

	UCF 101				HMDB 51				Youtube			
K	512	256	128	64	512	256	128	64	512	256	128	64
D3	84.35	84.32	83.03	81.34	56.14	55.29	54.71	51.70	89.91	91.55	91.09	90.36
VLAD	82.81	82.54	81.59	79.78	55.45	55.14	53.92	50.22	90.00	89.73	89.18	89.09
FV	85.23	84.82	83.80	82.48	58.13	57.34	55.88	53.20	91.00	91.00	90.73	90.45
D3+FV	**85.92**	**85.44**	**84.20**		**58.34**	**57.63**	**56.58**		**91.73**	**91.36**	90.45	
VLAD+FV	85.23	84.54	83.52		58.13	57.60	55.64		90.91	**91.36**	**90.82**	

including FV and VLAD+FV.

Two points are worth pointing out. First, the success of D3+FV shows that the information encoded in D3 and FV, although both used first and second order statistics of the two distributions, are complementary to each other. The hybrid of these two outperforms both D3 and FV. Since the running time of D3+FV is only roughly half of that of FV, D3+FV is attractive in both speed and accuracy. Second, VLAD+FV is obviously inferior to D3+FV. Its accuracy is very similar to that of FV, but lower than D3+FV in most cases.

3.3. Image recognition results

Now we test how D3 (and the comparison methods) work with instance vectors that are extracted by state-of-the-art deep learning methods. To extract instance vectors, we use the DSP (deep spatial pyramid) method [1], which spatially integrates deep fully convolutional networks. A set of instance vectors are efficiently extracted, each of which corresponds to a spatial region (*i.e.*, receptive field) in the original image. The CNN model we use is imagenet-vgg-verydeep-16 in [26] till the last convolutional layer, and the input image is resized such that its shortest edge is no smaller than 314 pixels, and its longest edge is no larger than 1120 pixels. Six spatial regions are used, corresponding to the level 1 and 0 regions in [29]. [1] finds that FV or VLAD usually achieves optimal performance with very small K sizes in DSP. Hence, we test $K \in \{4, 8\}$.

The following image datasets are used.

- Scene 15 [18]. It contains 15 categories of scene images. We use 100 training images per category, the rest are for testing.
- MIT indoor 67 [24]. It has 15620 images in 67 indoor scene types. We use the train/test split provided in [24].
- Caltech 101 [6]. It consists of 9K images in 101 object categories plus a background category. We train on 30 and test on 50 images per category.
- Caltech 256 [9]. It is a superset of Caltech 101, with 31K images, and 256 object plus 1 background categories. We train on 60 images per category, the rest for testing.
- SUN 397 [30]. It is a large scale scene recognition dataset, with 397 categories and at least 100 images per category. We use the first 3 train/test splits of [30].

Except for the indoor and SUN datasets, we run 3 random train/test splits in each dataset. Average accuracy rates on these datasets are reported in Table 2. As shown by the standard deviation numbers in Table 2, the deep learning instance vectors are stable and the standard deviations are small in most cases. Thus, we tested with 3 random train/test splits instead of more (*e.g.*, 5 or 10).

D3 and D3+FV have shown excellent results when combining with instance vectors extracted by deep nets. We have the following key observations from Table 2, which mostly coincides well with the observations concerning action recognition in Table 1. The last row in Table 2 shows the current state-of-the-art recognition accuracy in the literature, which are achieved by various systems that depend on deep learning using the same evaluation protocol.

D3 is slightly better than FV. D3 is better than FV in 3 datasets (Scene 15, indoor 67 and SUN 397), but worse than FV in the two Caltech datasets. It is worth noting that D3's accuracy is higher than that of FV by a larger margin in indoor 67 (1–2%) and SUN 397 (1.5–2.2%), while FV is only higher than D3 by 0.3–0.7% in the Caltech 101 and 256 datasets. Another important observation is that the win/loss are consistent among the train/test splits. In other words, if D3 wins (loses) in one dataset, it wins (loses) consistently in all three splits.[1] Thus, the CNN instance vectors lead to stable comparison results, and we believe 3 train/test splits are enough to compare these algorithms.

VLAD is better than both D3 and FV, but D3 bridges the gap between VLAD and FV. Although FV usually outperforms VLAD in image classification and retrieval using dense SIFT features and in the action recognition results of Table 1, a reversed trend is shown in Table 2 using CNN instance vectors. VLAD is almost consistently better than FV, up to 3.2% higher in the SUN 397 dataset. The accuracy of D3, however, is much closer to that of VLAD than FV's accuracy. D3 is usually 0.3%–0.6% lower than VLAD, with only two cases up to 1.1% ($K = 8$ in Caltech 256 and SUN

[1] Detailed per-split accuracy numbers are omitted.

Table 2. Image recognition accuracy (percent) comparisons. The definition of K is the same as that used in Table 1. The best results are shown in bold face. Standard deviations are also showed after the $\pm$ sign.

	Scene 15		MIT indoor 67		Caltech 101		Caltech 256		SUN 397	
	$K=4$	$K=8$	$K=4$	$K=8$	$K=4$	$K=8$	$K=4$	$K=8$	$K=4$	$K=8$
D3	92.34$\pm$0.23	92.10$\pm$0.65	77.31	77.76	93.60$\pm$0.17	93.80$\pm$0.58	83.15$\pm$0.15	82.92$\pm$0.09	59.93$\pm$0.24	60.22$\pm$0.07
VLAD	92.58$\pm$0.60	92.61$\pm$0.42	**77.61**	**78.13**	94.20$\pm$0.39	94.11$\pm$0.57	84.01$\pm$0.02	84.00$\pm$0.10	60.61$\pm$0.25	61.22$\pm$0.33
FV	91.96$\pm$0.40	91.53$\pm$0.56	75.97	75.82	94.32$\pm$0.51	94.10$\pm$0.33	83.75$\pm$0.16	83.40$\pm$0.13	58.40$\pm$0.12	57.97$\pm$0.28
D3+FV	**92.83**$\pm$0.55	92.82$\pm$0.31	77.09	77.99	**94.72**$\pm$0.51	**94.51**$\pm$0.44	**84.77**$\pm$0.12	**84.62**$\pm$0.15	**61.48**$\pm$0.22	61.38$\pm$0.52
VLAD+FV	92.82$\pm$0.52	92.76$\pm$0.56	77.54	78.06	94.71$\pm$0.41	94.45$\pm$0.51	84.18$\pm$0.51	84.61$\pm$0.16	61.32$\pm$0.26	**61.83**$\pm$0.27
D3+VLAD	92.82$\pm$0.30	**92.92**$\pm$0.19	77.01	77.91	94.59$\pm$0.54	94.45$\pm$0.41	84.09$\pm$0.25	84.31$\pm$0.14	60.38$\pm$0.30	61.48$\pm$0.32
	91.59$\pm$0.48 [34]		77.56 [8]		93.42$\pm$0.50 [10]		77.61$\pm$0.12 [2]		53.86$\pm$0.21 [34]	

397).

The hybrid methods are all effective, and D3+FV is the overall winning method. The second part of Table 2 presents results of hybrid methods. Beyond D3+FV and VLAD+FV, we also add the results of D3+VLAD, because VLAD is the winner in the first part of Table 2. Excluding the MIT indoor 67 dataset, obviously all hybrid methods have higher accuracy rates than every individual method. Among the hybrid methods, D3+FV is the overall winner again. It has the highest accuracy in 6 cases, while VLAD+FV and D3+VLAD has only one each. When comparing D3+FV with D3, FV or VLAD in detail, this hybrid method has higher accuracy than any single method in all train/test splits in all 36 comparisons (4 datasets excluding the indoor 67 dataset $\times$ 3 individual representations $\times$ 3 train/test splits). The MIT indoor 67 dataset is a special case, where VLAD is better than all other methods. We are not yet clear what characteristic of this dataset makes it particularly suitable for VLAD.

The fact that D3 is in general inferior to VLAD in this setup also indicates that CNN instance vectors have different characteristics than the dense SIFT vectors (cf. Sec. 3.1), for which VLAD is inferior to D3.

This might be caused by the fact that D3 and VLAD used very small K values ($K = 4$ or 8) with CNN instance vectors, compared to $K \geq 64$ in Sec. 3.1. Hence, both methods have much fewer number of dimensions now, and a few VLAD dimensions with highest discriminative powers may lead to better performance than D3. We will leave a careful, more detailed analysis of this observation to future work.

Significantly higher accuracy than state-of-the-art, especially in those difficult datasets. DSP [1] (with D3 or other individual representation methods) is a strong baseline, which already outperforms previous state-of-the-art in the literature (shown in the last row of Table 2). The hybrid method D3+FV leads to even better performance, *e.g.*, its accuracy is 7.2% higher than [2] for the Caltech 256 dataset,[2] and 7.6% higher than the place deep model of [34] for SUN 397.

[2] [26] reported an average recall rate of 86.2% for Caltech 256. DSP's average recall is 89.12% and D3+FV is 90.25% ($K = 4$).

3.4. Discussions

Overall, the proposed D3 representation method has the following properties:

- **D3 is discriminative, efficient, and *stable*.** D3 is not the individual representation method that leads to the highest accuracy. FV is the best in our action recognition experiments with ITF instance vectors, while VLAD is the best in our image categorization experiments using CNN features. It is, however, *the most stable one*. It is only slightly worse than FV in action recognition and slightly worse than VLAD in image categorization. Although VLAD is outperformed by FV by a large margin in action recognition (Table 1) and vice versa for image categorization (Table 2), D3 has stably achieved high accuracy rates. D3 is also as efficient as VLAD, and is much faster than the FV method;
- **D3+FV is the overall winning method.** Using the same number of dimensions for all individual and hybrid methods, D3+VLAD has shown the best performance, which indicates that the information encoded by D3 and FV form a synergy. Since the FV part of D3 only uses half the number of Gaussian components than that in individual FV, D3+FV is still more efficient than FV alone.

In short, D3 and D3+FV are effective and efficient in encoding entities that are represented as sets of instance vectors.

4. Conclusions and Future Work

We proposed the Discriminative Distribution Distance (D3) method to encode an entity (which comprises of a set of instance vectors) into a vector representation. Unlike existing methods such as FV, VLAD or Super Vectors that are designed from a generative perspective, D3 is based on discriminative ideas. We proposed to use directional distances to measure how two distributions (sets of vectors) are different with each other, and proposed to use the robust MPM classifier to robustly estimate this distance.

These discriminative design choices lead to excellent classification accuracy of the proposed D3 representation, which are verified by extensive experiments on action and

image categorization datasets. D3 is also efficient, and the hybrid D3+FV representation has achieved the best results among compared individual and hybrid methods.

In the same spirit as D3, we plan to combine D3 and FV in a principled way, which will add discriminative perspectives to FV and will further reduce the computational cost of the hybrid representation using D3+FV. We will further study how the benefits of VLAD can be utilized (*e.g.*, when CNN instance vectors are used).

References

[1] A. Author. Anonymized paper. In *Anonymous Conference*, Anonymous year. 7, 8

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 8

[3] M. Cimpoi, S. Maji, and A. Vedaldi. Deep convolutional filter banks for texture recognition and segmentation. In *CVPR*, 2015. 1

[4] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004. 1

[5] A. DasGupta. *Probability for statistics and machine learning: fundamentals and advanced topics*. Springer Science & Business Media, 2011. 3

[6] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training example: an incremental Bayesian approach tested on 101 object categories. In *CVPR 2004, Workshop on Generative-Model Based Vision*, 2004. 7

[7] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 1

[8] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, LNCS 8695, pages 392–407, 2014. 1, 8

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report CNS-TR-2007-001, Caltech, 2007. 7

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, volume LNCS 8691, pages 346–361, 2014. 8

[11] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS 11*, pages 487–493, 1999. 1

[12] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local images descriptors into compact codes. *IEEE TPAMI*, 34(9):1704–1716, 2012. 1

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678, 2014. 1

[14] Y.-G. Jiang, J. Liu, A. R. Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar. THUMOS: The first international workshop on action recognition with a large number of classes, 2013. 6

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS 25*, pages 1097–1105, 2012. 1

[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011. 6

[17] G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002. 4

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume II, pages 2169–2178, 2006. 6, 7

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[20] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, pages 1996–2003, 2009. 6

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[22] X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting VLAD with supervised dictionary learning and high-order statistics. In *ECCV*, LNCS 8691, pages 660–674, 2014. 5

[23] B. Póczos, L. Xiong, D. J. Sutherland, and J. Schneide. Nonparametric kernel estimators for image classification. In *CVPR*, pages 2989–2996, 2012. 4

[24] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, pages 413–420, 2009. 7

[25] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013. 1, 2

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7, 8

[27] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human action classes from videos in the wild. Technical report, CRCV-TR-12-01, University of Central Florida, 2012. 6

[28] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558, 2013. 1, 6

[29] J. Wu and J. M. Rehg. CENTRIST: A visual descriptor for scene categorization. *IEEE TPAMI*, 33(8):1489–1501, 2011. 1, 7

[30] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010. 7

[31] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, 2015. 1

[32] D. Yoo, S. Park, J.-Y. Lee, and I. S. Kweon. Fisher kernel for deep neural activations. *arXiv:1412.1628v2*, 2014. 1

[33] Y. Zhang, J. Wu, and J. Cai. Compact representation for image classification: To choose or to compress? In *CVPR*, pages 907–914, 2014. 6

[34] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using Places database. *NIPS 27*, pages 487–495, 2014. 8

[35] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using Super-Vector coding of local image descriptors. In *ECCV*, LNCS 6315, pages 141–154, 2010. 1